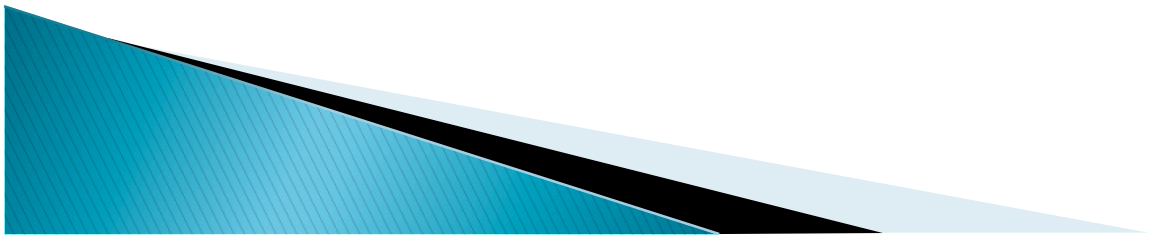
Review First-Order Logic

Zahra Mohammadi Zadeh

Sources: CS 157 lecture notes and B&L ch 2-4 slides.

KR & R and Logic

- Knowledge representation: symbolic encoding of propositions believed (by some agent) - Logic is a tool a "symbolic language" to represent knowledge



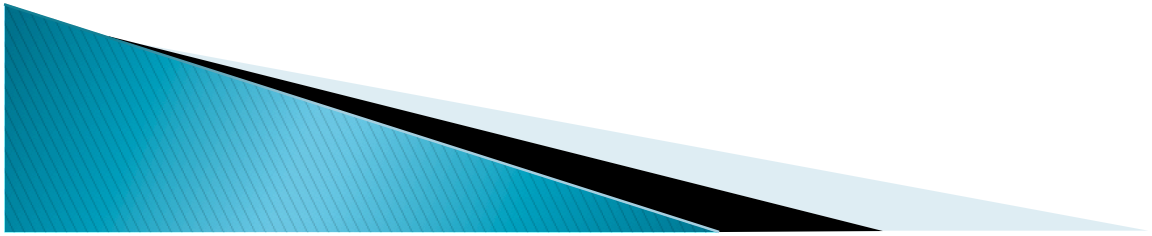
Using Logic

- No universal language / semantics
  - Why not English?
  - Different tasks / worlds
  - Different ways to carve up the world
- No universal reasoning scheme
  - Geared to language
  - Sometimes want "extralogical" reasoning
- Start with first-order predicate calculus (FOL)

  invented by philosopher Frege for the formalization of mathematics but will consider subsets / supersets and very different looking representation languages

Propositional Logic

- Concerned with propositions and their interrelationships
- Roughly speaking, a *proposition* is a possible condition of the world about which we want to say something. In other words, something that can be evaluated to true or false.
- p, q are propositions so are −p, p∧q, p∨q, p=>q (−p∨q) and p<=>q (p=>q ∧ q=>p)



Propositional Logic Semantics

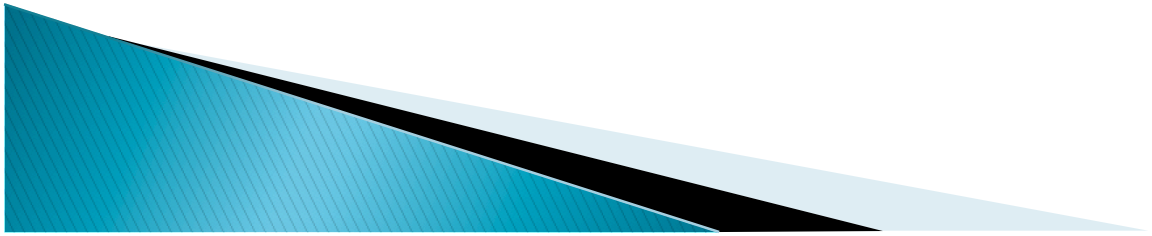
- Logical reasoning methods are designed to work no matter what meanings or values are assigned to the logical "variables" used in sentences. (ie. P by itself could be true or false).
- Although the values assigned to variables are not crucial in the sense just described, in talking *about* logic itself, it is sometimes useful to make these assignments explicit and to consider various assignments or all assignments and so forth. Such an assignment is called an interpretation.
- Formally, an *interpretation* for propositional logic is a mapping assigning a truth value to each of the simple sentences of the language.
- Example : for propositions p, q, r we define interpretation i such that p^i = true, q^i = false, r^i = true then (p=>q) is evaluated to false.

Prop. Logic Semantics continued

- Reverse Evaluation: can also generate all the interpretations (truth table for p, q, r in this case) and evaluate the propositional sentence under each case.
- Satisfiability: We say that an interpretation *i satisfies* a sentence if and only if it is *true* under that interpretation.
- Validity: A sentence is *valid* if and only if it is satisfied by *every* interpretation. (ex. p∨−p)
- Unsatisfiability: A sentence is *unsatisfiable* if and only if it is not satisfied by any interpretation. (ex. P <=>−p)

Entailment

Validity, satisfiability, and unsatisfiability are properties of individual sentences. In logical reasoning, we are not so much concerned with individual sentences as we are with the relationships between sentences. In particular, we would like to know, given some sentences (in KR the knowledge base), whether other sentences are or are not logical conclusions. This relative property is known as *logical entailment*.



Entailment continued

- Formally, a set of sentences Δ *logically entails* a sentence ϕ (written Δ |= ϕ) if and only if every interpretation that satisfies Δ also satisfies ϕ. For example, the set of sentences {*p,* p∧r*}* logically entails the sentence (*p* ∨ *q*).

- Inference: the process of calculating entailments

  sound: get only entailments

  complete: get all entailments

- Sometimes want unsound / incomplete reasoning

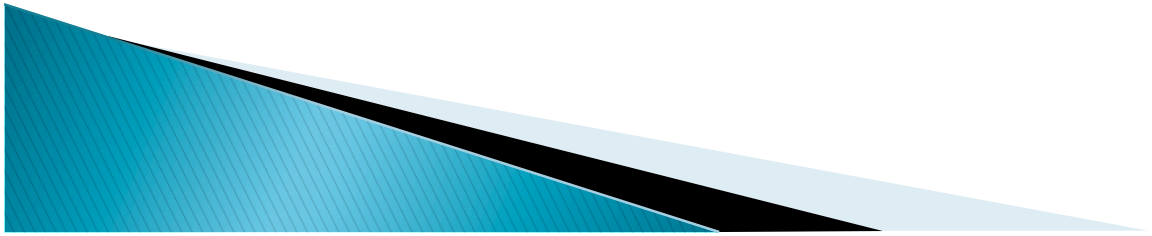
  for reasons to be discussed later

- Logic: study of entailment relations

Propositional Resolution

- Propositional resolution is a powerful rule of inference. (remember inference is the process of finding entailments which is an important task in KR&R)

- Clausal form: Propositional resolution works only on expressions in *clausal form*. Before the rule can be applied, the premises and conclusions must be converted to this form.

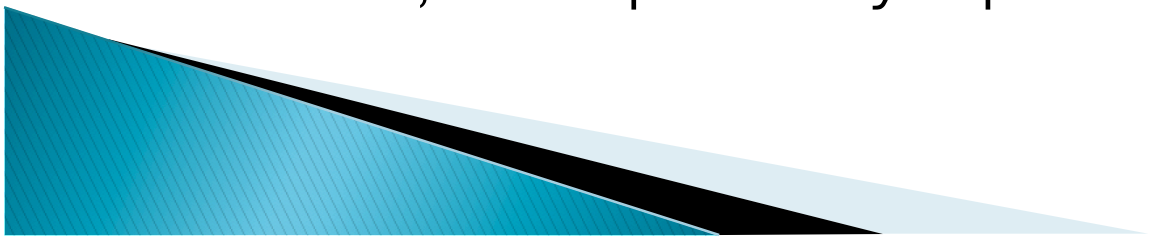


Clausal Form

- A *literal* is either an atomic sentence or a negation of an atomic sentence. For example, if p is a logical constant, p, $\neg p$ then are both literals.
- A *clause expression* is either a literal or a disjunction of literals.
- A *clause* is the set of literals in a clause expression. For example, the following sets are the clauses.

$$\{p\} \ , \{\neg p\}, \{\neg p, q\}$$

- Note that the empty set {} is also a clause. It is equivalent to an empty disjunction and, therefore, is unsatisfiable. As we shall see, it is a particularly important special case.



Conversion to Clausal from

- The conversion rules are summarized below and should be applied in order. (INDO)

1. Implications (I):

$$\phi \Rightarrow \psi \rightarrow \neg\phi \vee \psi$$

2. Negations (N):

$$\neg\neg\phi \rightarrow \phi$$

$$\neg(\phi \wedge \psi) \rightarrow \neg\phi \vee \neg\psi$$

3. Distribution (D):

$$\phi \vee (\psi \wedge \chi) \rightarrow (\phi \vee \psi) \wedge (\phi \vee \chi)$$

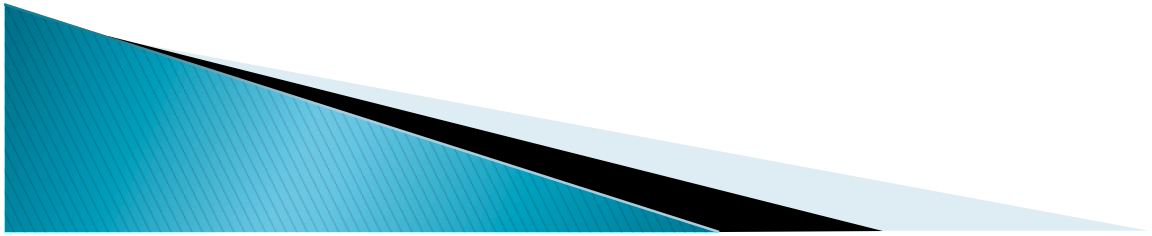
4. Operators (O):

$$\phi_1 \vee \ldots \vee \phi_n \rightarrow \{\phi_1, \ldots, \phi_n\}$$

$$\phi_1 \wedge \ldots \wedge \phi_n \rightarrow \{\phi_1\}, \ldots, \{\phi_n\}$$

Example

	$\neg(g \wedge (r \Rightarrow f))$
I	$\neg(g \wedge (\neg r \vee f))$
N	$\neg g \vee \neg(\neg r \vee f)$
	$\neg g \vee (\neg\neg r \wedge \neg f)$
	$\neg g \vee (r \wedge \neg f)$
D	$(\neg g \vee r) \wedge (\neg g \vee \neg f)$
O	$\{\neg g, r\}\ \{\neg g, \neg f\}$



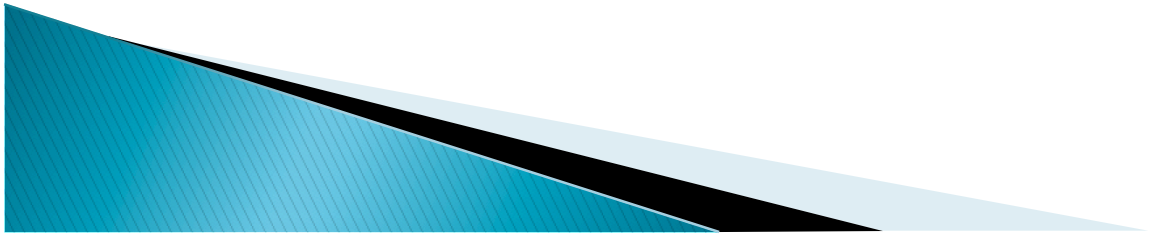
Resolution Step

- given a clause containing a literal χ and another clause containing the literal $\neg\chi$, we can infer the clause consisting of all the literals of both clauses without the complementary pair.

$$\{\phi_1, \ldots, \chi, \ldots, \phi_n\}$$

$$\{\psi_1, \ldots, \neg\chi, \ldots, \psi_n\}$$

$$\{\phi_1, \ldots, \phi, \psi_1, \ldots, \psi_n\}$$

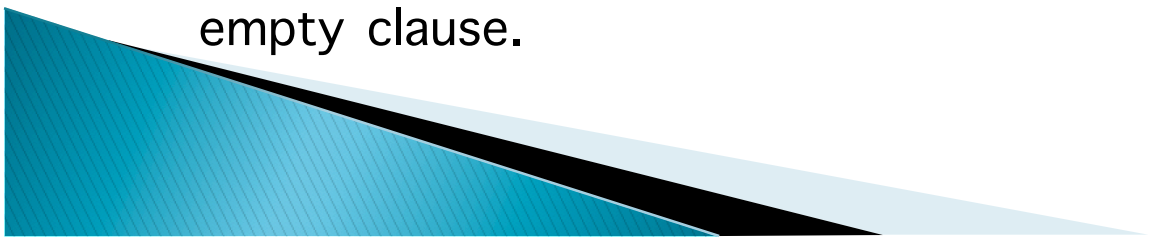


Resolution Application

- The problem is that propositional resolution is not *generatively* complete, i.e. we cannot directly derive all consequences from a set of premises. For example, we cannot derive tautology ($p \Rightarrow (q \Rightarrow p)$) directly using propositional resolution. There are no premises. Consequently, there are no conclusions.

- This apparent problem disappears if we take the clausal form of the premises (if any) together with the negated goal and try to derive the empty clause.

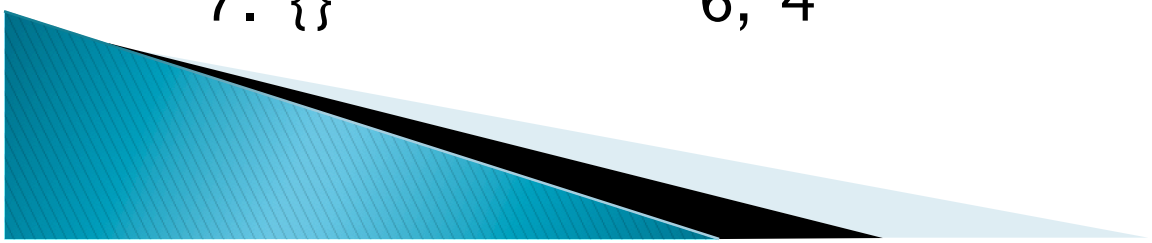
- To determine whether a set Δ of sentences logically entails a sentence ϕ, rewrite $\Delta \cup \{\neg\phi\}$ in clausal form and try to derive the empty clause.



Resolution Example

For the premises $(p \Rightarrow q)$ and $(q \Rightarrow r)$, we want to prove $(p \Rightarrow r)$.

1.	$\{\neg p, q\}$	Premise
2.	$\{\neg q, r\}$	Premise
3.	$\{p\}$	Negated Goal
4.	$\{\neg r\}$	Negated Goal
5.	$\{q\}$	3, 1
6.	$\{r\}$	5, 2
7.	$\{\}$	6, 4



First Order Logic

- Propositional Logic gives us the machinery to derive logical conclusions based on the relationship between propositions but this is not adequate.

- Relational Logic, provides us with a way of talking about individual objects and their interrelationships. It allows us to assert the existence of objects with a given relationship and allows us to talk about all objects having a given relationship.

Alphabet

- Logical symbols:

  Punctuation: (,), .

  Connectives: −,∧,∨, forall, exists, (=)

  Variables: *x, x1, x2, ..., x', x'', ..., y, ..., z, ...*

  - Fixed meaning and use like keywords in a programming language
  - Like variables in programming languages, the variables in FOL have a scope determined by the quantifiers
    - Lexical scope for variables

      P(x) ∧ exists. x[P(x) ∨ Q(x)] (first x is free and second one is bound)

- Non-logical symbols

  Predicate symbols (like Dog) **Note: not treating = as a predicate**

  Function symbols (like bestFriendOf)

  Domain-dependent meaning and use

  - like identifiers in a programming language

Grammar

Terms

1. Every variable is a term.
2. If *t1, t2, ..., tn are terms and f is a function of arity n,* then *f(t1, t2, ..., tn) is a term.*

Atomic wffs (well-formed formula)

1. If *t1, t2, ..., tn are terms and P is a predicate of arity n,* then *P(t1, t2, ..., tn) is an atomic wff.*
2. If *t1 and t2 are terms, then (t1=t2) is an atomic wff.*

Wffs

1. Every atomic wff is a wff.
2. If α and β are wffs, and v *is a variable, then* $-\alpha$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, *exists v.* α, *forall v.* α are wffs.

A sentence: wff with no free variables (closed)

The propositional subset: no terms, no quantifiers

- Atomic wffs: only predicates of 0-arity: $(p \wedge -(q \vee r))$

Semantics: Interpretation

- Logical interpretation: specification of how to understand predicate and function symbols
- Given a specification of
  - what objects there are
  - which of them satisfy *P*
  - what mapping is denoted by *f*

  it will be possible to say which sentences of FOL are true
- The FOL assumption:

  this is all you need to know about the non-logical symbols to understand which sentences of FOL are true or false

  remember the interpretations in propositional logic! We only need to know a truth assignment to determine the truth value of a logical sentence

Interpretations

Two parts: $\mathfrak{I} = \langle D, I \rangle$

D is the domain of discourse

can be *any* non-empty set

not just formal / mathematical objects

e.g. people, tables, numbers, sentences, unicorns, chunks of peanut butter, situations, the universe

I is an interpretation mapping

If P is a predicate symbol of arity n,

$I[P] \subseteq D \times D \times \ldots \times D$

an n-ary relation over D

If f is a function symbol of arity n,

$I[f] \in [D \times D \times \ldots \times D \rightarrow D]$

an n-ary function over D

for propositional symbols,

$I[p] = \{\}$ or $I[p] = \{\langle\rangle\}$

for constants, $I[c] \in D$

In propositional case, convenient to assume

$\mathfrak{I} = I \in [\textit{prop. symbols} \rightarrow \{\text{true, false}\}]$

Then Similar notion of satisfiability, unsatisfiability and entailment!

Basic Facts

Usually atomic sentences and negations

type facts

Man(john),

Woman(jane),

Company(faultyInsuranceCorp)

property facts

Rich(john),

¬HappilyMarried(jim),

WorksFor(jim,fic)

equality facts

john = ceoOf(fic),

fic = faultyInsuranceCorp,

bestFriendOf(jim) = john

Like a simple database (can store in a table)

Expressing Complex Facts

Universal abbreviations

$\forall y[\text{Woman}(y) \wedge y \neq \text{jane} \supset \text{Loves}(y,\text{john})]$

$\forall y[\text{Rich}(y) \wedge \text{Man}(y) \supset \text{Loves}(y,\text{jane})]$

$\forall x \forall y[\text{Loves}(x,y) \supset \neg\text{Blackmails}(x,y)]$

possible to express without quantifiers

Incomplete knowledge

$\text{Loves}(\text{jane},\text{john}) \vee \text{Loves}(\text{jane},\text{jim})$

which?

$\exists x[\text{Adult}(x) \wedge \text{Blackmails}(x,\text{john})]$

who?

cannot write down a more complete version

Closure axioms

$\forall x[\text{Person}(x) \supset x=\text{jane} \vee x=\text{john} \vee x=\text{jim} \ldots]$

$\forall x \forall y[\text{MarriedTo}(x,y) \supset \ldots]$

$\forall x[\, x=\text{fic} \vee x=\text{jane} \vee x=\text{john} \vee x=\text{jim} \ldots]$

limit the domain of discourse

also useful to have $\text{jane} \neq \text{john}$ ...

Expressing Terminological Facts

General relationships among predicates. For example:

disjoint $\forall x[\text{Man}(x) \supset \neg\text{Woman}(x)]$

subtype $\forall x[\text{Senator}(x) \supset \text{Legislator}(x)]$

exhaustive $\forall x[\text{Adult}(x) \supset \text{Man}(x) \vee \text{Woman}(x)]$

symmetry $\forall x \forall y\ [\text{MarriedTo}(x,y) \supset \text{MarriedTo}(y,x)]$

inverse $\forall x \forall y\ [\text{ChildOf}(x,y) \supset \text{ParentOf}(y,x)]$

type restriction $\forall x \forall y\ [\text{MarriedTo}(x,y) \supset \text{Person}(x) \wedge \text{Person}(y) \wedge \text{OppSex}(x,y)]$

sometimes

Usually universally quantified conditionals or biconditionals

Finally Proving the Entailment

Is there a company whose CEO loves Jane?

$\exists x\ [\text{Company}(x) \wedge \text{Loves}(\text{ceoOf}(x), \text{jane})]$??

Suppose $\mathfrak{I} \models$ KB.

Then $\mathfrak{I} \models \text{Rich}(\text{john}),\ \text{Man}(\text{john}),$

and $\mathfrak{I} \models \forall y[\text{Rich}(y) \wedge \text{Man}(y) \supset \text{Loves}(y, \text{jane})]$

so $\mathfrak{I} \models \text{Loves}(\text{john}, \text{jane}).$

Also $\mathfrak{I} \models \text{john} = \text{ceoOf}(\text{fic}),$

so $\mathfrak{I} \models \text{Loves}(\text{ceoOf}(\text{fic}), \text{jane}).$

Finally $\mathfrak{I} \models \text{Company}(\text{faultyInsuranceCorp}),$

and $\mathfrak{I} \models \text{fic} = \text{faultyInsuranceCorp},$

so $\mathfrak{I} \models \text{Company}(\text{fic}).$

Thus, $\mathfrak{I} \models \text{Company}(\text{fic}) \wedge \text{Loves}(\text{ceoOf}(\text{fic}), \text{jane}),$

and so

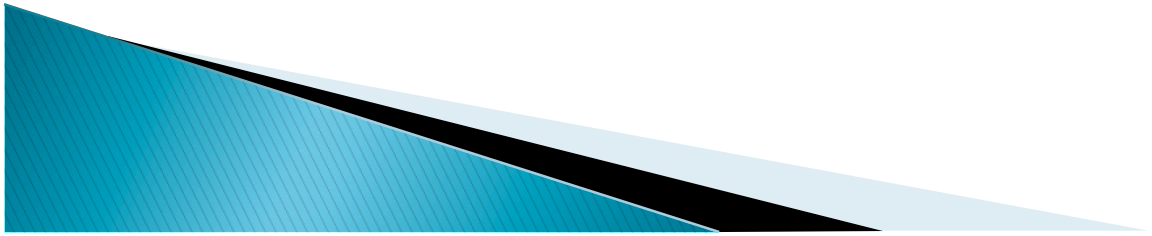
$\mathfrak{I} \models \exists x\ [\text{Company}(x) \wedge \text{Loves}(\text{ceoOf}(x), \text{jane})].$

Can extract identity of company from this proof

Summary

We covered:

- Propositional logic (as a special case of relational logic)
- Propositional resolution
- Relational logic
- I didn't talk about first-order resolution (see B&L chapter 4 and ask your questions)



Questions?